SIFT vs SURF comparison

Daniel Sjöholm

May 17, 2015

Abstract

This essay considers some differences and similarities between the SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-up Robust Features) algorithms, and compares them to each other in terms of speed and robustness. It starts out by briefly describing the point and need of feature matching, before starting the actual comparison. During the comparison SIFT and SURF are compared against each other in more detail, revealing an upper edge towards SURF in most aspects. An application for each of the algorithms is then suggested, where each algorithm will probably do a job better than its counterpart.

Introduction

Feature detection and matching is a big part of computer vision and is used for example in object recognition and visual navigation, helping for example to build up maps while navigating. A crucial point when matching is to be able to find robust matches, that will be detected as the same even when the viewpoint has changed or a luminance change has changed parts of the scene. For this same reason, detectors focusing on corners instead of edges are also more heavily used due to that corners are better localized than edges.

Two popular algorithms for feature detection and matching, are the algorithms Scale Invariant Feature Transform, or SIFT, and Speeded-up Robust Features (SURF). They have become popular for their speed of detection and high robustness in comparison to other methods. This essay is will compare the SIFT and SURF algorithms on a couple of points, specifically speed and robustness. The comparison is entirely based upon the following two papers, [2] for SIFT and [1] for the SURF algorithm as well as a small comparison between SIFT and SURF.

Comparison

One could directly from the names of the two algorithms get a hint at in which direction they are going to excel: *Scale Invariant ..* for SIFT and *Speeded-up Robust ..* for SURF, but the question is if the names do the algorithms justice when compared to each other, as SURF was developed after SIFT, and might because of that have a more fitting name for this comparison.

Robustness

Starting the comparison with robustness between the two algorithms, SIFT features are invariant to image scale and rotation, as well as to 3D viewpoint changes and changes in illumination. Whereas SURF is specifically built up around being invariant to scaling and in-plane rotations, as well as to contrast & illumination changes, but not to other 3D changes such as out of plane rotations. The authors to the article on SURF do however argue that perspective effects are assumed to be second order effects and partly covered by the descriptor robustness. The method of achieving the scale and rotation invariance is however different between the two algorithms.

SIFT uses a difference of gaussians approach, by convoluting the image with a gaussian kernel and then down-sampling the image and repeating it all. They then take the difference between two gaussian filtered images, which allows for feature detection in varying scales. The SIFT feature descriptor also takes into consideration that features could shift their position a bit, depending on scale or simple angle difference between pictures, and uses a method for allowing this without a large change in the descriptor. In addition to this, a sub algorithm is being used in order to sort out edge features, as they are not well localized inside an image and could disturb the detection of good matches. Features with low contrast are removed as well, since they could more easily have originated from noise.

SURF on the other hand uses an approach which doesn't require scaling of the underlying image, instead the image is converted into an integral image, which allows for fast box convolutions. The downside of this approach with fast box filters is that the jump between the first two scales of detection, is quite big (a factor of 1.4 to 1.7), which could possibly lead to some missing features.

SIFT is completely invariant to rotations, whereas SURF has weak point for rotations around odd multiples of 45°, where repeatability takes a 5-10% hit [1]. SIFT was also shown to perform very well even for large database sizes (100 000+ entries), and the repeatability of finding the correct match in the database only decreased marginally with increasing number of entries.

It feels like SIFT should be the more robust algorithm, considering the higher degree of invariance to various transformations, but what is presented in [1] states that SURF actually performs comparable in all cases, even outperforming in many, in terms of repeatably finding the correct match between several images. The authors believe that this is due to the fact that the SIFT gradients used for matching are more affected by high-frequency noise, whereas the SURF descriptors stay the same. Summarizing this, the descriptors of SURF actually seem to be more robust, even though no more specific steps are taken towards invariance for out-of-plane rotations for example.

Complexity and speed

SURF has a couple of speed improvements over SIFT. The most important speedup factor of SURF compared to SIFT, is the usage of integral images. This allows SURF to use box filters with very large filter sizes, and still have a constant time usage for the

filter step, independent of the filter size. This works thanks to how integral images are built up.

Integral images store the sum of all intensities starting from the orig (usually the top left most corner) up to a certain coordinate, as the value for that pixel. This allows intensity sums over a region to be computed using just 3 additions, independent of how large the region is. SURF uses the fast box convolutions to achieve scale invariance by increasing the filter-size instead of reducing the image dimensions, as thanks to the integral image, the convolutions can be performed with constant speed, independent of the filter size, something which is not possible with the difference of gaussians approach used by SIFT.

The difference of gaussians method requires more computation time, due to the down-sampling of images and re-filtering. Of course every new filter step takes shorter time due to less pixels to work on, but each step still contributes a non-insignificant part to the computational time.

The two algorithms have the same speed for brightness and contrast invariance however, as they just normalize the found descriptor before storing it.

The important step after having performed the feature matching, is to actually compare the found features to what's already stored in a database. SIFT uses descriptors that are 128 units long, whereas the selected feature length in [1] was 64 units. This has a speed impact as when comparing matches against the database it's going to take twice as long time for SIFT to compute the closest match in the database.

Application

Usage of these algorithms where one is better than the other leads me to suggest camera calibration for SURF (or U-SURF, depending on if you know that images are going to be free from large rotations) and SIFT for object recognition in clustered scenes. The reason behind this, is that SURF is very robust and still precise, something which is very important in order to get a good and reliable camera calibration, and outperforms SIFT in exactly this manner. SIFT on the other hand seems to be good at finding matching features even when there have been a lot of transformations between the two matches, and is highly invariant to rotations whereas SURF has weak points for rotations of odd multiples of 45°. This gives SIFT an advantage when it comes to detecting an object which can be found in a scene with various rotations, giving it a better recognition ability. SIFT is as well shown to perform well even for large databases of matches (100 000+ entries), showing a high repeatability that only decreases marginally with increasing database size. This is naturally important for object recognition, as once the database grows large you still need to be able to find the correct match.

Conclusion

In almost all aspects, SURF seems to be the better algorithm of the two. It could be that the paper used as a base comparison [1], is biased towards SURF, but the end effect

is that I'd choose SURF, or it's upright version U-SURF for most applications. This because of it's higher robustness, as well as it's shorter computation time by a factor of 3-5 compared to SIFT [1], without losing much precision in terms of finding good features.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[2] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.